**Cloude X**

**Architecture**

* **Env Provider**

환경변수에 관한 내용을 제공하고 인프라 자원에 대한 설정을 한다(OS, 미들웨어, 네트워크 등의 시스템 기반 자원에 대한 설정). Setting DB에 이러한 설정을 저장해 둔다.

* **Setting DB**

Env Provider가 제공한 내용을 저장해 두고 시스템이 필요로 할 때 제공.

* **API DB**

API에 관련된 자원들(Rest API에서 사용하는 개념과 유사한 듯). 예를 들어 Resource (URI pattern, name 등), Method, Parameter, 라우팅 관련 정보 등을 저장하는 DB.

* **Gateway**

외부 요청을 처리하여 관련 API로 라우팅하여 메시지 큐로 전달한다. 이후 처리된 결과를 가져와 Composition하는 역할 또한 수행하는 것으로 보인다.

* **Rabbit Message Queue**

AMQP 프로토콜을 구현한 메시지 브로커. 시스템 간에 메시지를 전달해 주는 서비스로 이해하면 편하다.

* **Commax Device Manager**

실질적으로 전달된 메시지를 처리하고 디바이스를 관리하는 어플리케이션. 디바이스 데이터 베이스에 기기를 등록하고 관리하는 등 관리 전반의 역할을 수행한다.

* **Device DB**

디바이스에 관한 정보가 담긴 DB 디바이스 등록 정보 및 기기의 상태 등을 저장하고 디바이스 관리에 사용하는 듯하다.

**Class Loader Hierarchy**

* **Launcher Class Loader**

어플리케이션이 실행될 때 메모리에 적재되어야 할 클래스들을 찾아 적재해주는 역할을 하는 Loader.

* **Bootstrap Class Loader**

JVM App Class를 적재하기 위한 명령어와 그 주소를 가지고 있다가 부팅시에 명령어들을 읽어와 하나씩 수행하며 JVM App Class를 메모리에 적재하고 실행 가능한 상태까지 끌어올리는 역할을 하는 Loader.

* **JVM App Class Loader**

JVM(Java Virtual Machine)에 어플리케이션 코드를 로딩 하여 적재하는 역할을 수행한다. JVM위에 코드를 적재하여 수행함으로써 OS환경에 영향을 받지 않고 수행이 가능하게 하는 Loader.

* **Core Class Loader**

앞서 동작들은 어플리케이션의 실행을 위해 어플리케이션 외적인 환경 부분과 어플리케이션의 단지 코어 부분만 JVM에 적재하기 위한 동작이었다. Core Class Loader는 어플리케이션의 실제 동작을 위한 코어 부분을 JVM에 적재하여 어플리케이션의 실행을 준비하는 Loader.

* **Shared Class Loader**

애플리케이션 간에 공유되어야 하거나 공유되어야 되는 Class를 적재하여 여러 실행 단위들이 공유하여 사용하게 함으로써 처리 속도와 메모리 사용을 줄일 수 있다. 이러한 공유되는 Class를 적재하는 역할을 하는 Loader.

* **Pocat Application Class Loader**

실제 어플리케이션의 실행 단위 코드를 실행하고 적재하는 Loader. 이러한 단위들 간의 공통부분 공유를 위해 Shared Class에서 호출하게 되어있는 듯하다.

**Class Loading 관계도**

위에서 살펴본 Application실행까지의 과정을 Class Loader 대신 Jar파일과 Container 그리고 Lib중심으로 정리해둔 흐름도.